Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	2	"20010052034".pn.	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/09/30 11:36
L2	2	"6289510".pn.	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/09/30 11:36
S1	2	"6154878".pn.	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/09/30 11:35
S2	1252	717/168-173.ccls.	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/10 07:10
S3	88	S2 and ((dynamic on$1line (on adj the adj2 fly) hot) with (replac$3 updat$3 patch$3) with (software$1 library libraries module$1 function$1))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 09:00
S4	3579	719/310,313-316,328-329, 331-332.ccls.	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 07:17
S5	6	S3 and S4	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 08:52
S6	5	(stand$1in adj2 object$1).ab.	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 07:24
S7	0	(stand$1in adj2 object$1).ab. and ((updat$1 upgrad$3) with software$1)	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 07:24

S8	2	"5732275".pn.	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 08:53
S9	2	"5764992".pn.	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 08:53
S10	2477	((dynamic on$1line (on adj the adj2 fly) hot) with (replac$3 updat$3 patch$3) with (software$1 library libraries module$1 function$1))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 09:01
S11	2790	((dynamic on$1line (on adj the adj2 fly) hot) with (replac$6 updat$3 patch$3) with (software$1 library libraries module$1 function$1))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 09:23
S12	15	S11 and ((track$3 count$3) with (enter$3 exit$3) with (software$1 module$1 library libraries))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 09:29
S13	304	S2 and ((track$3 count$3 updat$ increas$3 descreas$3) with (enter$3 exit$3 invok$3 call$3))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 09:30
S14	235	S13 and ((updat$3 upgrad$3 replac$5) with (software module$1 library libraries))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 09:14
S15	1087	"717"/$.ccls. and ((track$3 count$3 increas$3 descreas$3) with (enter$3 exit$3 invok$3 call$3))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 09:23
S16	39	S15 and ((dynamic on$1line (on adj the adj2 fly) hot) with (replac$6 updat$3 patch$3) with (software$1 library libraries module$1 function$1))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 10:42
S17	15	S11 and ((track$3 count$3 increas$3 descreas$3) with (enter$3 exit$3) with (software$1 module$1 library libraries))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 09:29

S18	340	((dynamic on$1line (on adj the adj2 fly) hot) with (replac$6 updat$3 patch$3) with (software$1 library libraries module$1 function$1)) and ((track$3 count$3 increas$3 descreas$3) with (enter$3 exit$3 invok$3 call$3))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 10:48
S19	48	S18 and ((count reference) near5 zero)	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 09:31
S20	220	"717"/$.ccls. and ((dynamic on$1line (on adj the adj2 fly) hot) with (replac$6 updat$3 patch$3) with (software$1 library libraries module$1 function$1))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 10:45
S21	11	S20 and ((re$1nam$3 (name near3 chang$3)) with (function$1 method$1))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 10:43
S22	28	((re$1nam$3 (name near3 chang$3)) with (function$1 method$1)) and ((dynamic on$1line (on adj the adj2 fly) hot) with (replac$6 updat$3 patch$3) with (software$1 library libraries module$1 function$1))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 10:45
S23	18	((dynamic on$1line (on adj the adj2 fly) hot) with (replac$6 updat$3 patch$3) with (software$1 library libraries module$1 function$1)) and (renam$3 with (function$1 method$1))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 11:00
S24	20	((dynamic on$1line (on adj the adj2 fly) hot) with (replac$6 updat$3 patch$3) with (software$1 library libraries module$1 function$1)) and (renam$3 with (function$1 method$1 module$1))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 11:01
S25	520	((replac$6 updat$3 patch$3) with (software$1 library libraries module$1 function$1)) and (renam$3 with (function$1 method$1 module$1))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 11:08

S26	22	((replac$6 updat$3 patch$3) with (software$1 library libraries module$1 function$1)).ab. and (renam$3 with (function$1 method$1 module$1))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 11:01
S27	82	S25 and "717"/$.ccls.	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 11:09
S28	32	S25 and "719"/$.ccls.	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 11:09
S29	1040	((replac$6 updat$3 patch$3) with (software$1 library libraries module$1 function$1)) and ((renam$3 (name near5 chang$3) (name near3 replac$3)) with (function$1 method$1 module$1))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 11:08
S30	151	S29 and "717"/$.ccls.	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 11:09
S31	72	S29 and "719"/$.ccls.	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/09 11:09
S32	83	(software with (updat$3 upgrad$3 replac$6)) and (re$1nam$3 near5 (method$1 function$1))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/10 07:02
S33	13	717/168-173.ccls. and (re$1nam$3 with (function$1 method$1))	US-PGPUB; USPAT; EPO; DERWENT; IBM_TDB	OR	OFF	2005/05/10 07:10

PORTAL USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: The ACM Digital Library The Guide

+"dynamic software updating" Search

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **dynamic software updating** Found **19** of **161,645**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 19 of 19

Relevance scale

1 Mutatis mutandis: safe and predictable dynamic software updating

Gareth Stoyle, Michael Hicks, Gavin Bierman, Peter Sewell, Iulian Neamtiu

January 2005 **ACM SIGPLAN Notices , Proceedings of the 32nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages**, Volume 40 Issue 1

Full text available: pdf(273.03 KB) Additional Information: full citation, abstract, references, index terms

Dynamic software updates can be used to fix bugs or add features to a running program without downtime. Essential for some applications and convenient for others, low-level dynamic updating has been used for many years. Perhaps surprisingly, there is little high-level understanding or language support to help programmers write dynamic updates effectively.To bridge this gap, we present Proteus, a core calculus for dynamic software updating in C-like languages that is flexible, safe, and predictab ...

Keywords: capability, dynamic software updating, proteus, type inference, updateability analysis

2 Dynamic software updating

Michael Hicks, Jonathan T. Moore, Scott Nettles

May 2001 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation**, Volume 36 Issue 5

Full text available: pdf(1.44 MB) Additional Information: full citation, abstract, references, citings, index terms

Many important applications must run continuously and without interruption, yet must be changed to fix bugs or upgrade functionality. No prior general-purpose methodology for dynamic updating achieves a practical balance between flexibility, robustness, low overhead, and ease of use.

We present a new approach for C-like languages that provides type-safe dynamic updating of native code in an extremely flexible manner (code, data, and types may be updated, at programmer-determined times ...

3 Mining Software Repositories (MSR): Understanding source code evolution using abstract syntax tree matching

Iulian Neamtiu, Jeffrey S. Foster, Michael Hicks

May 2005 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2005 workshop on Mining software repositories MSR '05**, Volume 30 Issue 4

Full text available: pdf(201.75 KB) Additional Information: full citation, abstract, references, index terms

Mining software repositories at the source code level can provide a greater understanding of how software evolves. We present a tool for quickly comparing the source code of different versions of a C program. The approach is based on partial abstract syntax tree matching, and can track simple changes to global variables, types and functions. These changes can characterize aspects of software evolution useful for answering higher level questions. In particular, we consider how they could be used ...

Keywords: abstract syntax trees, software evolution, source code analysis

4 Dynamic rebinding for marshalling and update, with destruct-time ?

Gavin Bierman, Michael Hicks, Peter Sewell, Gareth Stoyle, Keith Wansbrough

August 2003 **ACM SIGPLAN Notices , Proceedings of the eighth ACM SIGPLAN international conference on Functional programming**, Volume 38 Issue 9

Full text available: pdf(199.01 KB) Additional Information: full citation, abstract, references, citings, index terms

Most programming languages adopt static binding, but for distributed programming an exclusive reliance on static binding is too restrictive: dynamic binding is required in various guises, for example when a marshalled value is received from the network, containing identifiers that must be rebound to local resources. Typically it is provided only by ad-hoc mechanisms that lack clean semantics.In this paper we adopt a foundational approach, developing core dynamic rebinding mechanisms as extension ...

Keywords: distributed programming, dynamic binding, dynamic update, lambda calculus, marshalling, programming languages, serialisation

5 Mobile computing and applications (MCA): A declarative framework for adaptable applications in heterogeneous environments

P. Inverardi, F. Mancinelli, M. Nesi

March 2004 **Proceedings of the 2004 ACM symposium on Applied computing**

Full text available: pdf(242.30 KB) Additional Information: full citation, abstract, references

In this paper we present an approach for developing adaptable software applications. The problem we are facing is that of a (possibly mobile) user who wants to download and execute an application from a remote server. The user's hosting device can be of different kinds (laptops, personal digital assistants, cellular phones, communicators, etc.) with specific hardware and software capabilities. The problem is to be able to decide whether the user's current device characteristics are compatible wi ...

6 Middleware for mobility: Dynamically programmable and reconfigurable middleware services

Manuel Roman, Nayeem Islam

October 2004 **Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware**

Full text available: pdf(651.39 KB) Additional Information: full citation, abstract, references

The increasing software complexity and proliferation of distributed applications for cell phones demand the introduction of middleware services to assist in the development of advanced applications. However, from the user perspective, it is essential that these new phones provide a smooth error-free experience. Despite of the complexity underlying a cell phone, placing a phone call remains a simple task that can be performed by most users regardless of their technical background. Furthermore, ce ...

7 Predicting problems caused by component upgrades
Stephen McCamant, Michael D. Ernst
September 2003 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering**, Volume 28 Issue 5

Full text available: pdf(142.32 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a new, automatic technique to assess whether replacing a component of a software system by a purportedly compatible component may change the behavior of the system. The technique operates before integrating the new component into the system or running system tests, permitting quicker and cheaper identification of problems. It takes into account the system's use of the component, because a particular component upgrade may be desirable in one context but undesirable in another. No forma ...

Keywords: software components, software upgrades, specification matching

8 Posters: Dynamically updatable component-based system (DUCS)
Robert Bialek
October 2003 **Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**

Full text available: pdf(61.18 KB) Additional Information: full citation, abstract, references, index terms

We present our framework called DUCS. DUCS supports on-the-fly updates of distributed component-based applications. DUCS has a layered architecture running on top of a virtual machine extended with a support for object unload and replacement. DUCS is a meta-level framework, so the impact on the application's implementation is minimal. It supports dynamic component replacement, state transfer among components, interface modifications, and automatic propagation of updates. The framework is expanda ...

Keywords: components, dynamic updates, interface adaptation, reflection, state-transfer

9 Monitoring deployed software using software tomography
Jim Bowring, Alessandro Orso, Mary Jean Harrold
November 2002 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2002 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering**, Volume 28 Issue 1

Full text available: pdf(142.43 KB) Additional Information: full citation, abstract, references, citings, index terms

Software products are often released with missing functionality or errors that result in failures in the field. In previous work, we presented the Gamma technology, which facilitates remote monitoring of deployed software and allows for a prompt reaction to failures. In this paper, we investigate one of the principal technologies on which Gamma is based: software tomography. *Software tomography* splits monitoring tasks across many instances of the software, so that partia ...

Keywords: code coverage, gamma technology, software engineering, software tomography, tomographic refinement

10 Type-based hot swapping of running modules (extended abstract)
Dominic Duggan
October 2001 **ACM SIGPLAN Notices , Proceedings of the sixth ACM SIGPLAN**

international conference on Functional programming, Volume 36 Issue 10

Full text available: pdf(150.34 KB) Additional Information: full citation, abstract, references, citings, index terms

While dynamic linking has become an integral part of the run-time execution of modem programming languages, there is increasing recognition of the need for support for hot swapping of running modules, particularly in long-lived server applications. The interesting challenge for such a facility is to allow the new module to change the types exported by the original module, while preserving type safety. This paper describes a type-based approach to hot swapping running modules. The approach is bas ...

Keywords: dynamic typing, hot swapping, module interconnection languages, shared libraries



11 Gamma system: continuous evolution of software after deployment

Alessandro Orso, Donglin Liang, Mary Jean Harrold, Richard Lipton

July 2002 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2002 ACM SIGSOFT international symposium on Software testing and analysis**, Volume 27 Issue 4

Full text available: pdf(141.81 KB) Additional Information: full citation, abstract, references, citings

In this paper, we present the GAMMA system, which facilitates remote monitoring of deployed software using a new approach that exploits the opportunities presented by a software product being used by many users connected through a network. GAMMA splits monitoring tasks across different instances of the software, so that partial information can be collected from different users by means of light-weight instrumentation, and integrated to gather the overall monitoring information. ...



12 Devirtualizable virtual machines enabling general, single-node, online maintenance

David E. Lowell, Yasushi Saito, Eileen J. Samberg

October 2004 **Proceedings of the 11th international conference on Architectural support for programming languages and operating systems**, Volume 32 , 38 , 39 Issue 5 , 5 , 11

Full text available: pdf(174.01 KB) Additional Information: full citation, abstract, references, index terms

Maintenance is the dominant source of downtime at high availability sites. Unfortunately, the dominant mechanism for reducing this downtime, cluster rolling upgrade, has two shortcomings that have prevented its broad acceptance. First, cluster-style maintenance over many nodes is typically performed a few nodes at a time, mak-ing maintenance slow and often impractical. Second, cluster-style maintenance does not work on single-node systems, despite the fact that their unavailability during mainte ...

Keywords: availability, online maintenance, planned downtime, virtual machines



13 Software Engineering for Secure Systems (SESS) --- Building Trustworthy Applications: Leveraging architectural models to inject trust into software systems

Somo Banerjee, Chris A. Mattmann, Nenad Medvidovic, Leana Golubchik

May 2005 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2005 workshop on Software engineering for secure systems—building trustworthy applications SESS '05**, Volume 30 Issue 4

Full text available: pdf(247.67 KB) Additional Information: full citation, abstract, references

Existing software systems have become increasingly durable and their lifetimes have significantly lengthened. They are increasingly distributed and decentralized. Our dependence on them has grown tremendously. As such, the issues of *trustworthiness* and *security* have become prime concerns in designing, constructing, and evolving software

systems. However, the exact meanings of these concepts are not universally agreed upon, nor is their role in the different phases of the software d ...

14 Frontmatter (includes Activities, Conference Corner, and Technical Correspondence)
SIGPLAN Notices staff
August 2003 **ACM SIGPLAN Notices**, Volume 38 Issue 8
Full text available: pdf(629.55 KB) Additional Information: full citation

15 Lazy modular upgrades in persistent object stores
Chandrasekhar Boyapati, Barbara Liskov, Liuba Shrira, Chuang-Hue Moh, Steven Richman
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications**, Volume 38 Issue 11
Full text available: pdf(304.38 KB) Additional Information: full citation, abstract, references, index terms

Persistent object stores require a way to automatically upgrade persistent objects, to change their code and storage representation. Automatic upgrades are a challenge for such systems. Upgrades must be performed in a way that is efficient both in space and time, and that does not stop application access to the store. In addition, however, the approach must be modular: it must allow programmers to reason locally about the correctness of their upgrades similar to the way they would reason about r ...

Keywords: ownership types, presistent objects, software upgrades

16 HydroJ: object-oriented pattern matching for evolvable distributed systems
Keunwoo Lee, Anthony LaMarca, Craig Chambers
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications**, Volume 38 Issue 11
Full text available: pdf(311.06 KB) Additional Information: full citation, abstract, references, citings, index terms

In an evolving software system, components must be able to change independently while remaining compatible with their peers. One obstacle to independent evolution is the *brittle parameter problem*: the ability of two components to communicate can depend on a number of *inessential* details of the types, structure, and/or contents of the values communicated. If these details change, then the components can no longer communicate, even if the *essential* parts of the message remain ...

Keywords: HydroJ, XML, distributed systems, dynamic dispatch, object-oriented programming, pattern matching, semi-structured data, software evolution, ubiquitous computing

17 Impala: a middleware system for managing autonomic, parallel sensor systems
Ting Liu, Margaret Martonosi
June 2003 **ACM SIGPLAN Notices , Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming**, Volume 38 Issue 10
Full text available: pdf(684.33 KB) Additional Information: full citation, abstract, references, citings, index terms

Sensor networks are long-running computer systems with many sensing/compute nodes working to gather information about their environment, process and fuse that information, and in some cases, actuate control mechanisms in response. Like traditional parallel

systems, communication between nodes is of fundamental importance, but is typically accomplished via wireless transceivers. One further key attribute of sensor networks is that they are almost always long running systems, intended to operate i ...

Keywords: middleware system, sensor networks, software adaptation, software update

18 Using mobile code to create ubiquitous augmented reality

Kari J. Kangas, Juha Röning

March 2002 **Wireless Networks**, Volume 8 Issue 2/3

Full text available: pdf(239.77 KB) Additional Information: full citation, abstract, references, index terms

Augmented reality systems supplement reality by adding virtual objects into a real-world view. In this article, we describe a flexible mobile code approach for implementing ubiquitous, active, and mobile augmented reality systems. We will concentrate primarily on solving the problem of how to acquire the data for the virtual objects in a way that will be flexible and expandable enough to be used in ubiquitous computing. To clarify the concepts and to illustrate our current research status, we wi ...

Keywords: augmented reality, mobile code, mobile computing, ubiquitous computing